# AFRL
# DoD Shared Resource Center
# (DSRC)


# HP XC Opteron (FALCON)
# User's Guide

## March 06, 2009

# Release Notes

Issued May 12, 2006

Pioneer release

Issued October 1, 2007

Reviewed and Updated with minor changes

Issued June 19, 2006

Updated for Production release

Issued January 20, 2008

Reviewed and updated to reflect change from ASC MSRC to AFRL MSRC

Issued February 4, 2008

Corrected link to The AFRL MSRC Batch Use Policy

Issued March 6, 2009

Updated all references from AFRL MSRC to AFRL DSRC

Minor usage updates

Table of Contents

1.0 Introduction..... 1
1.1 Assumed Background of the Reader ..... 1
1.2 Hardware Overview..... 1
1.3 Accessing the System ..... 1
1.4 AFRL DSRC Connectivity ..... 1
1.5 AFRL DSRC Startup Files ..... 1
1.6 The Archive Command..... 2
1.7 Additional Information..... 2
2. 0 AFRL DSRC HP XC Opteron ..... 3
2.1 Hardware..... 3
2.2 File System Overview..... 3
2.3 Operating System..... 4
2.4 Available Software..... 4
3.0 Program Development ..... 5
3.1 Development Tools ..... 5
3.2 Parallel Processing..... 5
3.3 FORTRAN Programming ..... 7
3.4 C/C++ Programming ..... 8
3.5 Libraries..... 8
4.0 Running Jobs..... 10
4.1 Interactive Use ..... 10
4.2 Batch Use..... 10
5.0 Customer Service..... 16
5.1 Customer Service Center ..... 16
5.2 AFRL DSRC Support ..... 16
5.3 AFRL DSRC Website..... 16
Appendix. Usage Hints ..... 18
A.1 Runtime Considerations ..... 18
A.2 Files and Filespace ..... 18

# 1.0 Introduction

This document provides an overview and introduction to using the HP XC Opteron pioneer system, located at the Air Force Research Laboratory (AFRL) Major Shared Resource Center (MSRC). The AFRL DSRC is located at Wright-Patterson Air Force Base, near Dayton, Ohio.

This guide is intended to provide information so that customers who are familiar with the Linux operating system can create and run their own programs, as well as use existing application software on the HP XC Opteron system.

## 1.1 Assumed Background of the Reader

It is assumed that the reader of this guide has a firm grasp of the concepts required to use a Linux operating system and to program in either the C, C++ or FORTRAN 77/90/95 languages.

## 1.2 Hardware Overview

The HP XC Opteron system is a Distributed Memory system with 2 Central Processing Units (CPUs) per node. Each CPU is a 2.8 GigaHertz (GHz) AMD Opteron processor and has a 64 kilobyte (KB) Level 1 cache and 1 MB of Level 2 cache. Due to Linux Kernel and MPI memory management, users have access to a minimum of 2.9 GB of memory per node, depending on the number of MPI processes the user spawns.

The HP XC Opteron has a total of 2048 CPUs, divided into 1024 nodes with 2 CPUs per node. There are four interactive nodes having 4 CPUs each.

## 1.3 Accessing the System

The fully qualified hostname of the interactive system is falcon.afrl.hpc.mil.

Users are only permitted to login onto the interactive nodes of the system. The other nodes are for batch jobs only. Users submit their jobs on the front-end node, and the batch system will automatically start their jobs on the other systems based on the load of the system.

## 1.4 AFRL DSRC Connectivity

Since the HP XC Opteron is an integrated component of the AFRL DSRC, user files are Network File System (NFS) mounted from the AFRL DSRC High Availability File Server (HAFS) system to the system. When users log into a system, their home (**$HOME**) directory (which will be the current directory immediately after logging in) physically resides on the file server, but appears to be local to the system. The AFRL DSRC also supplies archival storage and visualization capabilities.

## 1.5 AFRL DSRC Startup Files

All users are provided a .cshrc and .login file in their home directory. These files reference standard setup files, maintained by the site administrators in a central location, which set up a standard environment for all MSRC users. These files should not be modified.

To set up specific information for your HP XC Opteron session, such as environment variables, path information, terminal information, or command aliases, place the appropriate commands and information into created files called .personal.cshrc and .personal.login. The standard startup files check your home directory for the existence of these files and executes them if found. Commands related to aliases, prompts and some environment variables should go into .personal.cshrc, while commands related to the type of terminal you are using should go into .personal.login. See Section 3 of this guide for more details on the computing environment and the AFRL DSRC User's Guide for more details on startup files.

## 1.6 The Archive Command

The archive system (**$ARCHIVE_HOME**) is mounted on the interactive node of the system. However, to improve transfer speeds for batch jobs, **$ARCHIVE_HOME** is not NFS-mounted to the batch nodes. To transfer files from **$ARCHIVE_HOME**, users will have to use un-kerberized rcp or the archive command.

The archive command is a recently added tool to the AFRL DSRC to help users with transferring files to and from **$ARCHIVE_HOME**. The basic syntax for the archive command is:

**archive get [getopts] file1 [file2 ...] archive put [putopts] file1 [file2 ...]**

More information on the archive command can be found online via the archive man page (man archive).

## 1.7 Additional Information

Much of the information presented in this document is available online through the man pages and is accessible by typing:

**man {command name}**

# 2.0 AFRL DSRC HP XC Opteron

This section details the hardware and software available on the HP XC Opteron and how they are currently configured.

## 2.1 Hardware

The HP XC Opteron has a total of 2048 CPUs available in batch. Each CPU is a 2.8 GHz AMD Opteron processor with a peak speed of 5.7 GFLOPS, providing a total capacity of approximately 11.64 TFLOPs. Each CPU has a primary data cache of 64 KB, a primary instruction cache of 64 KB and access to 2 GB of system memory. At this time, the system is divided into 1024 separate nodes. The system has a Distributed MultiProcessing architecture with 2048 CPUs , 4 GB of memory per node and a total of 96 TB of disk space. Due to Linux Kernel and MPI memory management, users have access to a minimum of 2.9 GB of memory per node, depending on the number of MPI processes the user spawns.

When logging in to falcon, the user will be placed onto one of 4 interactive nodes due to load-balancing. Each of these nodes has identical hardware specifications to the batch nodes. However, there are 4 CPUs per node on these interactive nodes as opposed to only 2 CPUs on the batch nodes and the CPUs on the interactive nodes are 2.4 GHz as opposed to the 2.8 GHz CPUs on the batch nodes.

## 2.2 File System Overview

Diskspace is subdivided into two areas:

- System space (i.e., /usr, /opt. )
- /workspace

### 2.2.1 Workspace

Batch jobs are required to run in a local filesystem, generically known as workspace. Each HPC system has high-speed, parallel filesystems for this purpose. Workspace on the HP XC Opteron totals 96 TB.

I/O on the /workspace filesystem is quicker than I/O on a file system mounted from the network (such as **$HOME** or **$ARCHIVE_HOME**). **/workspace** is intended for the temporary storage of data files needed for your application. This includes (but is not limited to) grid files, restart files, input files and output files. **$WORKDIR** is to be used rather than the /tmp and /usr/tmp directory areas to prevent possible system crashes.

**There is no quota on the amount of disk space you may use in /workspace, but a file scrubber is used to automatically remove old files to prevent it from becoming filled.**

The current policy for removing files is on the AFRL DSRC web page and is subject to change based on periodic reviews.

The $WORKDIR file system is NOT backed up. In the event of deletion or catastrophic media failure, files and data structures are lost. It is your responsibility to transfer files that need to be saved to a location that allows permanent storage such as $HOME or, preferably, $ARCHIVE_HOME. Due to space restrictions on your $HOME directory, it is highly recommended that you use $ARCHIVE_HOME for long-term file storage and backups.

## 2.3 Operating System

The HP XC Opteron runs a variation of Red Hat Enterprise Linux AS release 4

## 2.4 Available Software

Software currently available on the HP XC Opteron includes: the PGI and GNU FORTRAN,

C, and C++ compilers and MPI and many third party software packages.

# 3.0 Program Development

Program development in the HP XC Opteron computing environment is similar to that used in a typical UNIX environment. However, the user must take additional steps to utilize the multiple processors available.

## 3.1 Development Tools

The AFRL DSRC offers many tools to help users who write their own code develop, compile and debug their software.

The HP XC Opteron has the PGI® High-Performance Fortran, C and C++ compilers and the GNU Fortran, C and C++ compilers installed.

The AMD Core Math Library (ACML) is installed, which provides a collection of mathematical and scientific libraries including Basic Linear Algebra Subprograms (BLAS) levels 1, 2 and 3; LAPACK; Fast Fourier Transforms (FFTs); and convolutions.

To aid users in debugging their software applications, the AFRL DSRC provides the Totalview Debugger, PGI Debugger (pgdbg) and the GNU Debugger (gdb). To launch the Totalview Debugger or view its manpage, users will have to load the Totalview module, via the module load totalview/ default command. Once the totalview module is loaded, the command to launch Totalview is totalview.

The Totalview Debugger and GNU Debugger are symbolic source code debuggers that debug programs compiled by the PGI® C/C++ Compiler, the PGI® Fortran Compiler, and the GNU compilers (gcc and g++). For full source-level debugging, compile the source code with the compiler option that includes the symbol table information in the compiled executable file.

Documentation for these compilers, libraries, and tools is available online in the man page by executing a man on pgf77, pgf90, pgf95, pgcc, pgCC, gcc, g++, g77, acml, totalview (after loading the Totalview module), pgdbg and gdb.

## 3.2 Parallel Processing

Users may utilize multiple CPUs to execute their programs. The compilers are capable of creating parallel programs through the use of compiler directives and parallel standards such as Message Passing Interface (MPI) and OpenMP.

### 3.2.1 MPI

The goal of MPI is to develop a widely used standard for writing message-passing programs. As such, the interface attempts to establish a practical, portable, efficient and flexible standard for message passing.

Users can compile MPI programs on the HP XC Opteron using the mpicc, mpiCC, mpif77 and mpif90 commands. These commands are wrappers around the PGI or GCC compilers that automatically link the appropriate MPI libraries for users. To change the compiler from

the default PGI, use one of the following: MPI_CC, MPI_CXX, MPI_F77, MPI_F90 For MPI FORTRAN codes:

**mpif77 -o prog prog.f mpif90 -o prog prog.f mpif95 -o prog prog.f**

For MPI C/C++ codes:

**mpicc -o prog prog.c**

**mpiCC -o prog prog.c**

Please consult the man pages on the compiler for more information. To execute MPI code on the HP XC Opteron, users must use the mpirun command. When using in batch, the users must use this mpirun command: mpirun -srun -n {# of CPUs} {My

Program} When using interactively, the users must use this mpirun command:

**mpirun -np {# of CPUs} {My Program}**

More information on MPI can be obtained from: http://www.mpi-forum.org

### 3.2.2 OpenMP

OpenMP is a specification for a set of compiler directives, library routines and environment variables that can be used to specify shared memory parallelism in FORTRAN and C/C++ programs.

Creating an OpenMP program is done through OpenMP directives in the source code and by adding the -mp flag to your compile string using the PGI compilers. OpenMP is currently unsupported in the GNU compilers.

To run an OpenMP program, users must first tell the program how many threads (processors) to use. This is achieved through the OMP_NUM_THREADS environment variable. To set this variable, use the following command in csh:

**setenv OMP_NUM_THREADS x**

where x is the number of CPUs you wish to utilize.

For more information on OpenMP and its directives, please see the

following page: http://www.openmp.org

### 3.2.3 MPICH

MPICH is a freely available, portable implementation of MPI, the

standard for message-passing libraries and the HP XC Opteron supports MPICH 1.2.

Users can create MPICH applications using the mpicc.mpich, mpiCC.mpich, mpif77.mpich and mpif90.mpich.

These commands are wrappers around the PGI compilers that automatically link the appropriate MPICH libraries for users.

For MPICH FORTRAN codes:

**mpif77.mpich -o prog prog.f**

**mpif90.mpich -o prog prog.f**

For MPICH C/C++ codes:

**mpicc.mpich -o prog prog.c**

**mpiCC.mpich -o prog prog.c**

Please consult the man pages on the compiler for more information.

To execute MPICH application on the HP XC Opteron users must use the mpirun.mpich command.

When using in batch, the users must use this mpirun.mpich command:

**mpirun.mpich -srun -n {# of CPUs} {My Program}**

When using interactively, the users must use this mpirun.mpich command:

**mpirun.mpich -np {# of CPUs} {My Program}**

More information on MPI can be obtained from: http://www.mpi-forum.org

### 3.2.4 MPI/OpenMP Hybrid Code

Some users are experimenting with codes by implementing a hybrid of MPI and OpenMP calls. While the HP XC Opteron can run such code, there are some issues and limitations with doing so.

Because of the system configuration, hybrid applications are only able to run a two OpenMP and a single MPI process per node and up to 1024 nodes per job.

The syntax for the HP XC Opteron's mpirun command is similar to other platforms users are used to running on. Users must also set the OMP_NUM_THREADS variable, much like in the OpenMP section above.

When using in batch, the users must use this MPI/OpenMP command:

**setenv OMP_NUM_THREADS {Number of threads}**

**mpirun -srun -n {number of cpus} {My program}**

When using interactively, the users must use this MPI/OpenMP

command:

**setenv OMP_NUM_THREADS {Number of threads} mpirun -np {number**

**of cpus} {My program}**

## 3.3 FORTRAN Programming

The recommended FORTRAN compiler on the HP XC Opteron is the PGI FORTRAN compiler. The FORTRAN compiler commands are pgf77, pgf90 and pgf95. These optimizing compilers can generate a 64-bit and 32-bit code. Compiling a FORTRAN program on the HP XC Opteron is similar to compiling a program on a typical UNIX system.

**pgf77 -o prog prog.f pgf90 -o prog prog.f90 pgf95 -o prog prog.f95**

**g77 -o prog prog.f**

This command creates an executable called prog. The program is run by typing the program name at the system prompt.

**./prog**

Further optimization is available through the use of compiler flags and compiler directives.

Please check the pgf77, pgf90, pgf95 and g77 man pages for more details.

## 3.4 C/C++ Programming

The PGI and GNU C and C++ compilers are available on the HP XC Opteron. These compilers are capable of optimizing and parallelizing code. Compiling a C program on the HP XC Opteron is similar to compiling a C program on a typical UNIX system.

**pgcc -o prog prog.c**

**gcc -o prog prog.c**

Compiling a C++ program is also just as similar.

**pgCC -o prog prog.cpp**

**g++ -o prog prog.cpp**

These commands will create an executable program in a file called prog. The program is executed by entering

**./prog**

Further optimization is available through the use of compiler flags and compiler directives.

Please consult the pgcc or pgCC man pages for more details.

## 3.5 Libraries

### 3.5.1 Math and Science Libraries

The HP XC Opteron has ACML, a collection of mathematical and scientific libraries including BLAS levels 1, 2 and 3; LAPACK, a collection of solvers for dense linear algebra problems, including linear equations, linear least square problems, eigenvalue problems, and singular value decomposition problems; Fast Fourier Transforms (FFTs) and convolutions. Both single-threaded and multi-threaded routines are available, and select routines have been highly optimized to greatly improve performance. Users should use these library routines whenever possible.

This library is not automatically included in the link path. The user must specify the library when linking, as in the following examples.

**pgf77 -o prog -fpic prog.f -lacml pgcc -o prog -fpic prog.c -lacml**

**pgCC -o prog -fpic prog.c -lacml**

Please consult the ACML web page, http://developer.amd.com/ acml.aspx or /app2/ pgi/linux86_64/VERSION/doc/acml.pdf on falcon for more details.

# 4.0 Running Jobs

## 4.1 Runtime Considerations

The HP XC Opteron allocates more resources to batch jobs than for interactive use. Users will obtain the best throughput for long running or large memory jobs by submitting jobs to the batch queues. Thus, batch use is recommended.

Interactive use is allowed, particularly for program development, including debugging and performance improvement, job preparation, job submission, and the preprocessing and post-processing of data. Four nodes on the system are available for interactive use and interactive jobs are limited to 4 CPUs with 15 minutes of CPU time per process. Jobs with larger resource requirements must be submitted to the batch queues.

When submitting a job, choose the smallest queue that accomodates the job's time and memory requirements. Jobs that request significantly more resources than are actually needed can result in longer wait times and inefficient use of the machine.

The workspace filesystems are local to the HP XC Opteron. Although $HOME is NFS-mounted internally to the compute nodes, I/O access from workspace will be faster than from the HAFS. Thus, it is recommended to keep I/O local to the system.

Below is a sample script that copies two input files (one from the $HOME directory and one from the archival storage system), a program to the user's workspace area, changes the workspace, runs the program, copies two output files (one to the $HOME directory and one

to the archival storage system) and then deletes the remaining files.

cd $WORKDIR cp $HOME/small.input
$WORKDIR archive get big.input
$WORKDIR archive get prog
$WORKDIR prog archive put big.output
mv small.output $HOME

rm small.input big.input prog big.ouput

## 4.2 Batch Use

Load Sharing Facility (LSF), a networked subsystem for submitting, monitoring and controlling a work load of batch jobs on one or more systems, is the batch system for the system. It provides services to monitor queue activity and to delete queued or running jobs. In the event of an orderly system shutdown, LSF jobs will be rerun from the beginning of the job, if specifically marked. More information about LSF is available at

http://www.platform.com/products/HPC/

To allow users to run longer in the queues, a 2 week (336 hour) limit has been implemented. However, to keep the queue structure fair to all users, several restrictions have been put into

place:

- The primary resource to schedule jobs by is the CPH hour (CPH). This quantity is equal to the wall time requested, multiplied by the number of CPUs, and cannot exceed a value of

50,000 per user. (ncpus * walltime) = CPH

Table 1: Example CPH Values

- A user can use, at most, 1000 CPUs for MPI jobs and MPI/OpenMP hybrid and 2 CPUs for SMP and MPI/OpenMP hybrid jobs.
- The maximum wall time of any queue shall not exceed 168 hours.
- A background job cannot start if there is a foreground job in any queue.

The list of queues and the upper limits of job resources for these queues are available on the web at

http://www.afrl.hpc.mil/overall/policy_procedure/policies/use_policy.php

These limits are subject to change based on the periodic review of system utilization and system configuration.

TIP: Because of CPH, it is not recommended that users accept the queue default walltime. If your job requires less time to run than the queue default, requesting the smaller of the two will result in a lower CPH and will allow you to run more jobs and reduce your queue wait time.

Example: User Joe submits a 4 CPU job using the queue default walltime of 336 hours. This results in a CPH of 1344. User Bob submits the same 4 CPU job, but only requests the 120 hours he needs to finish the job. His CPH would be 480. This allows user Bob to submit twice (2.8 to be exact) as many jobs as Joe in the same amount of CPH. If user Bob can alter his walltime and fit his jobs within 110 hours, he can submit 3 times the number of jobs as Joe. This also has a domino effect on the queue structure, resulting in faster throughput throughout the entire system.

### 4.2.1 Queueing Structure

The AFRL DSRC HP XC Opteron has three queues: debug, standard, and background. These queues are available 24 hours a day, 7 days a week.

The debug queue accepts jobs that require up to 32 CPUs, 1 hour of CPU time, and 32 GB of memory. This queue is intended for short

runs.

The standard queue is available for production work. Jobs that are submitted without any queue specified will go to the standard queue. Once submitted to this queue, jobs are assigned priority based upon project status and wait time in the queue.

The background queue is also available for production work. Jobs run in the background queue are not charged against a user's allocation. However, jobs in the background queue are only started when utilization of the machine is low and never when foreground jobs are waiting.

The list of queues and the upper limits of job resources for these queues are available

on the web at

http://www.afrl.hpc.mil/overall/policy_procedure/policies/use_policy.php

These limits are subject to change based on periodic review of system utilization and

system configuration.

### 4.2.2 Preparing Jobs

Before a user submits a job, he/she should prepare a job script. A job script is a UNIX shell script that contains all the commands the user will execute during the job. LSF will place the error and output files in the directory the job was submitted from, therefore, scripts must be written with this in mind. Here is a sample job script:

#Change to WORK_DIR directory and copy input file. # cd $WORK_DIR archive get -C {directory in $ARCHIVE_HOME} {filename} # #Run the analysis. # {My Program} # #Archive output and remove $WORK_DIR # tar cvf ../{output filename}.tar . archive put -C {directory in $ARCHIVE_HOME} ../{output filename}.tar rm -rf $WORK_DIR # #Exit the script. # exit

This script copies an input file and a program to the user's $WORK_DIR directory. The $WORK_DIR directory is a directory created by LSF for users to run their batch jobs. This directory has certain protections from the workspace scrubber. As long as the job is running, plus five days after it finishes, this directory will not be removed. Then the script changes to the $WORK_DIR directory, runs the program, copies two output files to permanent storage (one to the

$HOME directory and one to the archival storage system $ARCHIVE_HOME), and then deletes the remaining files from the the $WORK_DIR directory and exits.

NOTE: $WORK_DIR only exists in LSF, you will not be able to change to that direc-

tory using the variable $WORK_DIR.

### 4.2.3 Submitting Jobs

Once a job script is prepared, the bsub command is used to submit the script to LSF. The command has the following syntax:

bsub < script

Some important LSF options used are (type man bsub for a complete list of options available):

JOBID 3373 3971

When a job is submitted to LSF, a unique identifier is assigned to the job by the batch system similar to below:

2079.falcon-1.afrl.hpc.mil

This identifier is needed when deleting a job.

Options of bsub commands are specified within the script file itself. The options are specified using syntax similar to PBS, but each line that contains an option must begin with the #BSUB string. Options that are specified within the script file must precede the first executable shell command of the file as in the following example.

#!/bin/csh #BSUB -q standard #BSUB -n 1 #BSUB -W 168:00 #BSUB -J test #BSUB -o test.out #BSUB -e test.out #BSUB -a MPI #BSUB -P WP+WPASC00000000**

**This is an example number. To find your account number, check your $ACCOUNT variable using

echo $ACCOUNT

More sample batch scripts can be found at the following URL:

http://www.afrl.hpc.mil/customer/userdocs/samples/samplebatch.php

### 4.2.4 Monitoring Jobs

The bjobs command is used to report the status of the batch jobs that are currently queued or running. Type man bjobs for information about bjobs and the options that are available.

The bjobs command lists all jobs that are running and queued, as shown in the following example.

**% bjobs -u all**

| JOBID | USER | STAT | QUEUE | FROM_HOST | EXEC_HOST | JOBNAME | SUBMIT_TIME |
|---|---|---|---|---|---|---|---|
| 3373 | user1 | PEND | standard | falcon1041 | | test | Feb 5 15:22 |
| 3971 | user2 | RUN | standard | falcon1042 | 2*ls-host.l | test | Feb 5 15:22 |

Here is an explanation of the fields in the bjobs output.

**Table 2: Fields from bjobs**

| Item | Meaning |
|---|---|
| JOBID | A unique identifier that consists of the original request number and the machine from which the request was submitted. Format is nnn, where nnn is an integer. |
| USER | Username of person submitting the job. |
| STAT | Job status. “RUN” indicates the job is running; “PEND” indicates the job is queued. |

| Item | Meaning |
|---|---|
| QUEUE | Name of the queue where the job is waiting or executing. |
| FROM_HOST | Cluster domain from which the job was submitted. |
| EXEC_HOST | Cluster domain where the job is running. |
| JOBNAME | Name of the job. This is either the name of the script file submitted to LSF or the name chosen with the -J flag. |
| SUBMIT_TIME | The date and time the job was submitted |

### 4.2.5 Deleting Jobs

In LSF, queued or running jobs are removed using the bkill command. The syntax is

bkill request-id where request-id is the LSF identifier number. Example:

**JOBID 3373 USER user STAT PEND QUEUE standard FROM_**

**HOST falcon1041 EXEC_HOST 2*ls-host.l JOBNAME test SUBMIT_TIME**

**Feb 5 15:22**

**bkill 3373**

# 5. Customer Service

## 5.1 Consolidated Customer Assistance Center (CCAC)

For customer assistance, call the Consolidated Customer Assistance Center (CCAC) at 1-877-CCAC-039 (1-877-222-2039), (937) 255-0679, or DSN 785-0679, or send an e-mail with a description of the problem to help@ccac.hpc.mil.

If you have any questions regarding the AFRL DSRC, contact CCAC first. If your problem or question is beyond the scope of the CCAC, they will refer you to the appropriate resource.

## 5.2 AFRL DSRC Support

In-depth technical inquiries and problems are forwarded to the AFRL DSRC Customer Assistance and Technology Center (CATC), which pursues such inquiries and problems through resolution as rapidly as possible. The AFRL DSRC CATC will attempt to determine the nature of the problem, then identify and coordinate whatever resources are needed to resolve the problem.

## 5.3 AFRL DSRC Website

The AFRL DSRC website is the best source for current AFRL DSRC information. To access the AFRL DSRC website simply access this URL: http://www.afrl.hpc.mil.

Some of the topics found on the website include:

APPLICATIONS

Short and long descriptions of current AFRL DSRC applications http://www.afrl.hpc.mil/software/

SYSTEMS

Information on AFRL DSRC servers and Archival Storage

http://www.afrl.hpc.mil/hardware/

CONSOLIDATED CUSTOMER ASSISTANCE CENTER (CCAC)

Available CCAC Services

http://www.ccac.hpc.mil

ONLINE DOCUMENTATION

Listings of the AFRL DSRC User Guides are available for viewing. Instructions are given on obtaining postscript versions.

http://www.afrl.hpc.mil/customer/userdocs/

TRAINING

Current course offerings and schedule

https://okc.erdc.hpc.mil/index.jsp

FREQUENTLY ASKED QUESTIONS

Submit questions and read about various topics (such as “Customizing Your Environment”).

http://www.afrl.hpc.mil

## POLICIES AND PROCEDURES

The latest policies regarding usage of the AFRL DSRC resources http://www.afrl.hpc.mil/overall/policy_procedure/